UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/765,145	01/28/2004	Eun Hye Choi	248156US2RD	9722

22850 7590 09/22/2008

OBLON, SPIVAK, MCCLELLAND MAIER & NEUSTADT, P.C.
1940 DUKE STREET
ALEXANDRIA, VA 22314

EXAMINER	
LE, MIRANDA	

ART UNIT	PAPER NUMBER
2169	

NOTIFICATION DATE	DELIVERY MODE
09/22/2008	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

patentdocket@oblon.com
oblonpat@oblon.com
jgardner@oblon.com

Office Action Summary	Application No.	Applicant(s)	
	10/765,145	CHOI ET AL.	
	Examiner	Art Unit	
	MIRANDA LE	2169	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *12 May 2008*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1 and 4-21* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1 and 4-21* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

This communication is responsive to Amendment, filed 05/12/08.

Claims 1, 4-21 are pending in this application. This action is made Final.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

This application currently names joint inventors. In considering patentability of the claims under 35 U.S.C. 103(a), the examiner presumes that the subject matter of the various claims was commonly owned at the time any inventions covered therein were made absent any evidence to the contrary. Applicant is advised of the obligation under 37 CFR 1.56 to point out the inventor and invention dates of each claim that was not commonly owned at the time a later invention was made in order for the examiner to consider the applicability of 35 U.S.C. 103(c) and potential 35 U.S.C. 102(e), (f) or (g) prior art under 35 U.S.C. 103(a).

Claims 1, 4-6, 8-13, 15, 16, 18-21 are rejected under 35 U.S.C. 103(a) as being unpatentable over Holenstein et al. (US Patent No. 7,103,586), in view of Schrader et al. (US Patent No. 5,903,881).

As to claims 1, 19, 20, Holenstein teaches a concurrency control method in a transaction processing system for processing a plurality of transactions in parallel with respect to data, the concurrency control method comprising *(i.e. Database replication systems replicate blocks of transaction steps or operations with synchronous replication, and perform dual writes with queuing and blocking of transactions. Tokens are used to prepare a target database for replication from a source database and to confirm the preparation. Database replication systems switch between a synchronous replication mode and an asynchronous replication mode, and then back to a synchronous replication mode, based on detection of selected events, Summary)*:

producing a copy of data at a time of starting an access to the data by each transaction *(i.e. Filtering--The operation of selectively choosing rows or transactions to replicate, col. 5, lines 45-46)*;

judging whether a collision between one of reading access or writing access to be made by a first transaction with respect to a copy of data for the first transaction and another one of the reading access or writing access made by the second transaction with respect to a coy of the data for the second transaction will occur or not *(i.e. Another preferred embodiment of the present invention addresses collisions when all or some of the nodes of a bidirectional replication system temporarily switch from synchronous replication to asynchronous replication and subsequently switch back. When the system is switched to the asynchronous mode, collisions may occur among queued transactions. To address this issue, the following steps are performed upon detection that the synchronous replication mode may be restored, col. 26, lines 55-63)*, when the

first transaction and the second transaction are accepted at the same time as concurrent transactions for accessing the same location of the data, wherein the first transaction is started earlier than the second transaction *(i.e. Switching Between Synchronous and Asynchronous Replication Modes. Data replication systems are normally set to operate in either a synchronous or asynchronous replication mode. Synchronous systems are prone to failure due to a disruption in communication between nodes. Accordingly, a synchronous system may be programmed to automatically revert to an asynchronous system if such a failure is detected, col. 25, lines 22-52)*;

carrying out a processing for avoiding the collision due to the concurrent transactions *(i.e. Switching Between Synchronous and Asynchronous Replication Modes. Data replication systems are normally set to operate in either a synchronous or asynchronous replication mode. Synchronous systems are prone to failure due to a disruption in communication between nodes. Accordingly, a synchronous system may be programmed to automatically revert to an asynchronous system if such a failure is detected, col. 25, lines 22-52)* when the judging step judges that the collision will occur *(i.e. Send the queued transactions that do not cause any collision to the other nodes in their order of occurrence. For example, transactions that caused collisions can be identified by comparing a unique record indicia (such as a unique primary key) across the nodes; those that were updated on multiple nodes during the asynchronous period have collided, col. 26, lines 64 to col. 27, line 3)*; and

reflecting a writing access made by the first transaction with respect to a copy of the data for the first transaction, on the data, when the first transaction is to be finished normally, and reflecting the writing access made by the first transaction also on a copy of the data for the second transaction if the second transaction is not finished yet *(i.e. a) Pick a winner based on some indicia of the change, such as most or least recent timestamp or sequence number, or pre-select a winner based on node information, such as the node location, node size, or node resources. b) As described above, use "relative" change information to determine the "final" value for the data that collided, and assign the final value to the data in both nodes. For example, if a part quantity initially starts at 100, and one node changes the part quantity from 100 to 30 (a net reduction of 70), while another changes the part quantity from 100 to 90 (a net reduction of 10), assigning a final value of 20 (100-70-10=100-10-70=20) to the part quantity on both nodes resolves the collision, col. 27, lines 7-21; See Fig. 8)*;

wherein a third copy of data is provided in which all the transactions having been finished are reflected *(i.e. a) Pick a winner based on some indicia of the change, such as most or least recent timestamp or sequence number, or pre-select a winner based on node information, such as the node location, node size, or node resources. b) As described above, use "relative" change information to determine the "final" value for the data that collided, and assign the final value to the data in both nodes. For example, if a part quantity initially starts at 100, and one node changes the part quantity from 100 to 30 (a net reduction of 70), while another changes the part quantity from 100 to 90 (a net*

reduction of 10), assigning a final value of 20 (100-70-10=100-10-70=20) to the part quantity on both nodes resolves the collision, col. 27, lines 7-21; See Fig. 8); and

when the first transaction is to make the reading access with respect to a copy of the data, the judging step judges whether the collision will occur or not according to whether first data looked up by making the reading access with respect to the copy of the data for the first transaction and second data looked up by making the reading access with respect to the third copy are identical or not *(i.e. a) Pick a winner based on some indicia of the change, such as most or least recent timestamp or sequence number, or pre-select a winner based on node information, such as the node location, node size, or node resources. b) As described above, use "relative" change information to determine the "final" value for the data that collided, and assign the final value to the data in both nodes. For example, if a part quantity initially starts at 100, and one node changes the part quantity from 100 to 30 (a net reduction of 70), while another changes the part quantity from 100 to 90 (a net reduction of 10), assigning a final value of 20 (100-70-10=100-10-70=20) to the part quantity on both nodes resolves the collision, col. 27, lines 7-21; See Fig. 8).*

Holenstein does not explicitly teach hierarchical data limitation.

Schrader teaches this limitation *(i.e. The database module 1407 is an interface to the user's data, which is stored in the transaction database. Records may be fetched or saved by the database module 1407 and this module allows all the other modules to access the transaction database. The database module 1403 preferably stores the user's data in combined relational-hierarchical data model, though other data models*

may also be employed. A hierarchical model is used to organize accounts per financial institution, such that all transactions for each account are stored with the account. A relational model is used for individual transactions, to provide associations between payees, amounts, and individual transactions within each account. Data that is stored includes payment data, transactions data, funds transfer data, and e-mail data. Transaction records may be variable length. Each record in the transaction data is marked as being either in the online statement 150, the mini checkbook 181, or the out box 167, and as being either reconciled or unreconciled. This marking allows for subsequent auto-reconciliation and reporting. Some transactions in the online statement 150 may be unreconciled if the user did not enter them first in the mini checkbook 181, col. 13, line 65 to col. 14, line 19).

It would have been obvious to one of ordinary skill of the art having the teaching of Holenstein at the time the invention was made to modify the system of Holenstein to include the limitations as taught by Schrader. One of ordinary skill in the art would be motivated to make this combination in order to store the user's data in combined relational-hierarchical data model in view of Schrader (col. 13, line 14 to col. 14, line 19), as doing so would give the added benefit of a hierarchical model using to organize accounts per financial institution, such that all transactions for each account are stored with the account as taught by Schrader (col. 13, line 65 to col. 14, line 19).

As per claim 21, Holenstein teaches a computer program product which employs a storage medium for causing a computer to function as a transaction

processing system for processing a plurality of transactions in parallel with respect to data, the computer program product comprising:

a first computer program code loaded in a processor for causing the computer to accept transactions which are temporarily overlapping *(See Fig. 8; The Shadowbase replication engine queues transaction steps or operations and then sends them in blocks of data to the other nodes for replication. That is, one block of data typically consists of a plurality of transaction steps or operations, col. 21, lines 30-62)*;

a second computer program code loaded in a processor for causing the computer to judge whether or not the copy of a first transactions will conflict with the copy of a second transaction after execution of the second transaction, wherein the first transaction is started earlier than the second transaction *(i.e. At the sub-record (i.e., field or sub-field level), collisions may not occur if they do not update the same field data in the record. For example, if one node modifies FIELD1 of a record ("record 1") while another node simultaneously updates FIELD2 of the same record ("record 1"), a collision does not occur as long as the granularity of the change that is applied to the other database is at the field or sub-field level. This form of collision resolution is acceptable as long as the combination of the new fields (new FIELD1 and FIELD2 values), along with the remaining record data, does not violate a referential integrity constraint or other business rule, col. 24, lines 14-26);*

a fourth computer program code loaded in a processor for causing the computer, when it is judged that a conflict will occurs between the first and second transactions *(i.e. Send the queued transactions that do not cause any collision to the other nodes in*

their order of occurrence. For example, transactions that caused collisions can be identified by comparing a unique record indicia (such as a unique primary key) across the nodes; those that were updated on multiple nodes during the asynchronous period have collided, col. 26, lines 64 to col. 27, line 3), to halt the execution of the second transaction until the execution of the first transaction has been completed *(i.e. Switching Between Synchronous and Asynchronous Replication Modes. Data replication systems are normally set to operate in either a synchronous or asynchronous replication mode. Synchronous systems are prone to failure due to a disruption in communication between nodes. Accordingly, a synchronous system may be programmed to automatically revert to an asynchronous system if such a failure is detected, col. 25, lines 22-52)*; and

a fifth computer program code loaded in a processor for causing the computer to reflect the copy of the first transaction in the data, when the first transactions is finished normally, and reflect the copy of the first transaction in the copy of the second transaction, when the second transaction is not finished yet *(i.e. a) Pick a winner based on some indicia of the change, such as most or least recent timestamp or sequence number, or pre-select a winner based on node information, such as the node location, node size, or node resources. b) As described above, use "relative" change information to determine the "final" value for the data that collided, and assign the final value to the data in both nodes. For example, if a part quantity initially starts at 100, and one node changes the part quantity from 100 to 30 (a net reduction of 70), while another changes the part quantity from 100 to 90 (a net reduction of 10), assigning a final value of 20*

(100-70-10=100-10-70=20) to the part quantity on both nodes resolves the collision, col. 27, lines 7-21; See Fig. 8);

wherein the second computer program code is executed to further produce a third copy of the data in which all the transactions having been finished are reflected *(i.e. a) Pick a winner based on some indicia of the change, such as most or least recent timestamp or sequence number, or pre-select a winner based on node information, such as the node location, node size, or node resources. b) As described above, use "relative" change information to determine the "final" value for the data that collided, and assign the final value to the data in both nodes. For example, if a part quantity initially starts at 100, and one node changes the part quantity from 100 to 30 (a net reduction of 70), while another changes the part quantity from 100 to 90 (a net reduction of 10), assigning a final value of 20 (100-70-10=100-10-70=20) to the part quantity on both nodes resolves the collision, col. 27, lines 7-21; See Fig. 8)*; and

when the first transaction is to make the reading access with respect to a copy of the data, the third computer program code is executed to judge whether the collision will occur or not according to whether first data looked up by making the reading access with respect to the copy of the data for the first transaction and second data looked up by making the reading access with respect to the third copy are identical or not *(i.e. a) Pick a winner based on some indicia of the change, such as most or least recent timestamp or sequence number, or pre-select a winner based on node information, such as the node location, node size, or node resources. b) As described above, use "relative" change information to determine the "final" value for the data that collided, and*

assign the final value to the data in both nodes. For example, if a part quantity initially starts at 100, and one node changes the part quantity from 100 to 30 (a net reduction of 70), while another changes the part quantity from 100 to 90 (a net reduction of 10), assigning a final value of 20 (100-70-10=100-10-70=20) to the part quantity on both nodes resolves the collision, col. 27, lines 7-21; See Fig. 8).

Holenstein does not specifically teach hierarchical data limitation.

Schrader teaches this limitation *(i.e. The database module 1407 is an interface to the user's data, which is stored in the transaction database. Records may be fetched or saved by the database module 1407 and this module allows all the other modules to access the transaction database. The database module 1403 preferably stores the user's data in combined relational-hierarchical data model, though other data models may also be employed. A hierarchical model is used to organize accounts per financial institution, such that all transactions for each account are stored with the account. A relational model is used for individual transactions, to provide associations between payees, amounts, and individual transactions within each account. Data that is stored includes payment data, transactions data, funds transfer data, and e-mail data. Transaction records may be variable length. Each record in the transaction data is marked as being either in the online statement 150, the mini checkbook 181, or the out box 167, and as being either reconciled or unreconciled. This marking allows for subsequent auto-reconciliation and reporting. Some transactions in the online statement 150 may be unreconciled if the user did not enter them first in the mini checkbook 181, col. 13, line 65 to col. 14, line 19).*

It would have been obvious to one of ordinary skill of the art having the teaching of Holenstein at the time the invention was made to modify the system of Holenstein to include the limitations as taught by Schrader. One of ordinary skill in the art would be motivated to make this combination in order to store the user's data in combined relational-hierarchical data model in view of Schrader (col. 13, line 14 to col. 14, line 19), as doing so would give the added benefit of using a hierarchical model to organize accounts per financial institution, such that all transactions for each account are stored with the account as taught by Schrader (col. 13, line 65 to col. 14, line 19).

As per claim 4, Holenstein teaches the concurrency control method of claim 3, wherein the judging step judges that the collision will not occur when the first data and the second data are judged as identical for all transactions that can be the second transaction, and judges that the collision will occur otherwise *(i.e. At the sub-record (i.e., field or sub-field level), collisions may not occur if they do not update the same field data in the record. For example, if one node modifies FIELD1 of a record ("record 1") while another node simultaneously updates FIELD2 of the same record ("record 1"), a collision does not occur as long as the granularity of the change that is applied to the other database is at the field or sub-field level. This form of collision resolution is acceptable as long as the combination of the new fields (new FIELD1 and FIELD2 values), along with the remaining record data, does not violate a referential integrity constraint or other business rule, col. 24, lines 14-26).*

As per claim 5, Holenstein teaches the concurrency control method of claim 1, further comprising: making the writing access with respect to a shared copy produced by copying the data in order to reflect writing accesses made by all transactions that make accesses to the data, when the first transaction is to make the writing access with respect to a copy of the data *(i.e. a) Pick a winner based on some indicia of the change, such as most or least recent timestamp or sequence number, or pre-select a winner based on node information, such as the node location, node size, or node resources. b) As described above, use "relative" change information to determine the "final" value for the data that collided, and assign the final value to the data in both nodes. For example, if a part quantity initially starts at 100, and one node changes the part quantity from 100 to 30 (a net reduction of 70), while another changes the part quantity from 100 to 90 (a net reduction of 10), assigning a final value of 20 (100-70-10=100-10-70=20) to the part quantity on both nodes resolves the collision, col. 27, lines 7-21; See Fig. 8)*;

wherein when the first transaction is to make the reading access with respect to a copy of the data, the judging step judges whether the collision will occur or not according to whether first data looked up by making the reading access and second data looked up by making the reading access with respect to the shared copy of the data are identical or not *(i.e. a) Pick a winner based on some indicia of the change, such as most or least recent timestamp or sequence number, or pre-select a winner based on node information, such as the node location, node size, or node resources. b) As described above, use "relative" change information to determine the "final" value for the data that collided, and assign the final value to the data in both nodes. For example,*

if a part quantity initially starts at 100, and one node changes the part quantity from 100 to 30 (a net reduction of 70), while another changes the part quantity from 100 to 90 (a net reduction of 10), assigning a final value of 20 (100-70-10=100-10-70=20) to the part quantity on both nodes resolves the collision, col. 27, lines 7-21; See Fig. 8).

Schrader teaches hierarchical data *(i.e. The database module 1407 is an interface to the user's data, which is stored in the transaction database. Records may be fetched or saved by the database module 1407 and this module allows all the other modules to access the transaction database. The database module 1403 preferably stores the user's data in combined relational-hierarchical data model, though other data models may also be employed. A hierarchical model is used to organize accounts per financial institution, such that all transactions for each account are stored with the account, col. 13, line 65 to col. 14, line 19).*

As per claim 6, Holenstein teaches the concurrency control method of claim 5, wherein the judging step judges that the collision will not occur when the first data and the second data are judged as identical, and judges that the collision will occur when the first data and the second data are judged as not identical *(i.e. a) Pick a winner based on some indicia of the change, such as most or least recent timestamp or sequence number, or pre-select a winner based on node information, such as the node location, node size, or node resources. b) As described above, use "relative" change information to determine the "final" value for the data that collided, and assign the final value to the data in both nodes. For example, if a part quantity initially starts at 100, and*

one node changes the part quantity from 100 to 30 (a net reduction of 70), while another changes the part quantity from 100 to 90 (a net reduction of 10), assigning a final value of 20 (100-70-10=100-10-70=20) to the part quantity on both nodes resolves the collision, col. 27, lines 7-21; See Fig. 8).

As per claim 8, Holenstein teaches the concurrency control method of claim 1, wherein when the first transaction is to make the writing access with respect to a copy of the data, the judging step judges whether the collision will occur or not according to whether first data looked up by making the reading access of the second transaction and second data looked up by making the reading access of the second transaction with respect to a state of the data after the writing access are identical or not, for all reading accesses by all transactions that make accesses to the data and that can be the second transaction *(i.e. a) Pick a winner based on some indicia of the change, such as most or least recent timestamp or sequence number, or pre-select a winner based on node information, such as the node location, node size, or node resources. b) As described above, use "relative" change information to determine the "final" value for the data that collided, and assign the final value to the data in both nodes. For example, if a part quantity initially starts at 100, and one node changes the part quantity from 100 to 30 (a net reduction of 70), while another changes the part quantity from 100 to 90 (a net reduction of 10), assigning a final value of 20 (100-70-10=100-10-70=20) to the part quantity on both nodes resolves the collision, col. 27, lines 7-21; See Fig. 8).*

Schrader teaches hierarchical data *(i.e. The database module 1407 is an interface to the user's data, which is stored in the transaction database. Records may be fetched or saved by the database module 1407 and this module allows all the other modules to access the transaction database. The database module 1403 preferably stores the user's data in combined relational-hierarchical data model, though other data models may also be employed. A hierarchical model is used to organize accounts per financial institution, such that all transactions for each account are stored with the account, col. 13, line 65 to col. 14, line 19).*

As per claim 9, Holenstein teaches the concurrency control method of claim 8, wherein the judging step judges that the collision will not occur when the first data and the second data are judged as identical for all reading accesses of all transactions that make accesses to the data and that can be the second transaction, and judges that the collision will occur otherwise *(i.e. a) Pick a winner based on some indicia of the change, such as most or least recent timestamp or sequence number, or pre-select a winner based on node information, such as the node location, node size, or node resources. b) As described above, use "relative" change information to determine the "final" value for the data that collided, and assign the final value to the data in both nodes. For example, if a part quantity initially starts at 100, and one node changes the part quantity from 100 to 30 (a net reduction of 70), while another changes the part quantity from 100 to 90 (a net reduction of 10), assigning a final value of 20 (100-70-10=100-10-70=20) to the part quantity on both nodes resolves the collision, col. 27, lines 7-21; See Fig. 8).*

Schrader teaches hierarchical data *(i.e. The database module 1407 is an interface to the user's data, which is stored in the transaction database. Records may be fetched or saved by the database module 1407 and this module allows all the other modules to access the transaction database. The database module 1403 preferably stores the user's data in combined relational-hierarchical data model, though other data models may also be employed. A hierarchical model is used to organize accounts per financial institution, such that all transactions for each account are stored with the account, col. 13, line 65 to col. 14, line 19).*

As per claim 10, Holenstein teaches the concurrency control method of claim 8, further comprising: recording an access sequence of accesses made with respect to a copy of the data by each transaction, for each one of all transactions that make accesses to the data *(i.e. a) Pick a winner based on some indicia of the change, such as most or least recent timestamp or sequence number, or pre-select a winner based on node information, such as the node location, node size, or node resources. b) As described above, use "relative" change information to determine the "final" value for the data that collided, and assign the final value to the data in both nodes. For example, if a part quantity initially starts at 100, and one node changes the part quantity from 100 to 30 (a net reduction of 70), while another changes the part quantity from 100 to 90 (a net reduction of 10), assigning a final value of 20 (100-70-10=100-10-70=20) to the part quantity on both nodes resolves the collision, col. 27, lines 7-21; See Fig. 8)*;

wherein the judging step obtains all reading accesses of all transactions that make accesses to the data and that can be the second transaction, by looking up a record of the access sequence *(i.e. a) Pick a winner based on some indicia of the change, such as most or least recent timestamp or sequence number, or pre-select a winner based on node information, such as the node location, node size, or node resources. b) As described above, use "relative" change information to determine the "final" value for the data that collided, and assign the final value to the data in both nodes. For example, if a part quantity initially starts at 100, and one node changes the part quantity from 100 to 30 (a net reduction of 70), while another changes the part quantity from 100 to 90 (a net reduction of 10), assigning a final value of 20 (100-70-10=100-10-70=20) to the part quantity on both nodes resolves the collision, col. 27, lines 7-21; See Fig. 8).*

Schrader teaches hierarchical data *(i.e. The database module 1407 is an interface to the user's data, which is stored in the transaction database. Records may be fetched or saved by the database module 1407 and this module allows all the other modules to access the transaction database. The database module 1403 preferably stores the user's data in combined relational-hierarchical data model, though other data models may also be employed. A hierarchical model is used to organize accounts per financial institution, such that all transactions for each account are stored with the account, col. 13, line 65 to col. 14, line 19).*

As per claim 11, Holenstein teaches the concurrency control method of claim 8, further comprising: recording data looked up by making the reading accesses *(i.e. a) Pick a winner based on some indicia of the change, such as most or least recent timestamp or sequence number, or pre-select a winner based on node information, such as the node location, node size, or node resources. b) As described above, use "relative" change information to determine the "final" value for the data that collided, and assign the final value to the data in both nodes. For example, if a part quantity initially starts at 100, and one node changes the part quantity from 100 to 30 (a net reduction of 70), while another changes the part quantity from 100 to 90 (a net reduction of 10), assigning a final value of 20 (100-70-10=100-10-70=20) to the part quantity on both nodes resolves the collision, col. 27, lines 7-21; See Fig. 8)*;

wherein the judging step obtains the first data by looking up a record of the data looked up *(i.e. a) Pick a winner based on some indicia of the change, such as most or least recent timestamp or sequence number, or pre-select a winner based on node information, such as the node location, node size, or node resources. b) As described above, use "relative" change information to determine the "final" value for the data that collided, and assign the final value to the data in both nodes. For example, if a part quantity initially starts at 100, and one node changes the part quantity from 100 to 30 (a net reduction of 70), while another changes the part quantity from 100 to 90 (a net reduction of 10), assigning a final value of 20 (100-70-10=100-10-70=20) to the part quantity on both nodes resolves the collision, col. 27, lines 7-21; See Fig. 8).*

As per claim 12, Holenstein teaches the concurrency control method of claim 8, wherein the judging step obtains the first data as data obtained by making the writing access that was made by the second transaction before the reading access, with respect to a state of the data at a start of the second transaction, and then making the reading access with respect to a state of the data after the writing access *(i.e. a) Pick a winner based on some indicia of the change, such as most or least recent timestamp or sequence number, or pre-select a winner based on node information, such as the node location, node size, or node resources. b) As described above, use "relative" change information to determine the "final" value for the data that collided, and assign the final value to the data in both nodes. For example, if a part quantity initially starts at 100, and one node changes the part quantity from 100 to 30 (a net reduction of 70), while another changes the part quantity from 100 to 90 (a net reduction of 10), assigning a final value of 20 (100-70-10=100-10-70=20) to the part quantity on both nodes resolves the collision, col. 27, lines 7-21; See Fig. 8).*

Schrader teaches hierarchical data *(i.e. The database module 1407 is an interface to the user's data, which is stored in the transaction database. Records may be fetched or saved by the database module 1407 and this module allows all the other modules to access the transaction database. The database module 1403 preferably stores the user's data in combined relational-hierarchical data model, though other data models may also be employed. A hierarchical model is used to organize accounts per financial institution, such that all transactions for each account are stored with the account, col. 13, line 65 to col. 14, line 19).*

As per claim 13, Holenstein teaches the concurrency control method of claim 8, further comprising:

making the writing access with respect to a shared copy produced by copying the data in order to reflect writing accesses made by all transactions that make accesses to the data, when the first transaction is to make the writing access with respect to a copy of the data *(i.e. a) Pick a winner based on some indicia of the change, such as most or least recent timestamp or sequence number, or pre-select a winner based on node information, such as the node location, node size, or node resources. b) As described above, use "relative" change information to determine the "final" value for the data that collided, and assign the final value to the data in both nodes. For example, if a part quantity initially starts at 100, and one node changes the part quantity from 100 to 30 (a net reduction of 70), while another changes the part quantity from 100 to 90 (a net reduction of 10), assigning a final value of 20 (100-70-10=100-10-70=20) to the part quantity on both nodes resolves the collision, col. 27, lines 7-21; See Fig. 8)*; and

storing states of the shared copy at timings at which the writing accesses were made by some of the transactions that make accesses to the data *(i.e. a) Pick a winner based on some indicia of the change, such as most or least recent timestamp or sequence number, or pre-select a winner based on node information, such as the node location, node size, or node resources. b) As described above, use "relative" change information to determine the "final" value for the data that collided, and assign the final value to the data in both nodes. For example, if a part quantity initially starts at 100, and one node changes the part quantity from 100 to 30 (a net reduction of 70), while another*

changes the part quantity from 100 to 90 (a net reduction of 10), assigning a final value of 20 (100-70-10=100-10-70=20) to the part quantity on both nodes resolves the collision, col. 27, lines 7-21; See Fig. 8);

wherein the judging step obtains the first data as data obtained by reproducing a state of the data at a timing at which the reading access was made by selecting one of stored states of the shared copy which is close to the state of the data at a timing at which the reading access was made and making the writing access that was made by the second transaction with respect to a selected state of the shared copy according to need, and then making the reading access with respect to a reproduced state of the data *(i.e. a) Pick a winner based on some indicia of the change, such as most or least recent timestamp or sequence number, or pre-select a winner based on node information, such as the node location, node size, or node resources. b) As described above, use "relative" change information to determine the "final" value for the data that collided, and assign the final value to the data in both nodes. For example, if a part quantity initially starts at 100, and one node changes the part quantity from 100 to 30 (a net reduction of 70), while another changes the part quantity from 100 to 90 (a net reduction of 10), assigning a final value of 20 (100-70-10=100-10-70=20) to the part quantity on both nodes resolves the collision, col. 27, lines 7-21; See Fig. 8).*

Schrader teaches hierarchical data *(i.e. The database module 1407 is an interface to the user's data, which is stored in the transaction database. Records may be fetched or saved by the database module 1407 and this module allows all the other modules to access the transaction database. The database module 1403 preferably*

stores the user's data in combined relational-hierarchical data model, though other data models may also be employed. A hierarchical model is used to organize accounts per financial institution, such that all transactions for each account are stored with the account, col. 13, line 65 to col. 14, line 19).

As per claim 15, Holenstein teaches the concurrency control method of claim 8, wherein the judging step obtains the second data as data obtained by making the writing access of the second transaction with respect to a state after the writing access was made with respect to a copy of the data for the first transaction; and then making the reading access with respect to a state of the data after the writing access of the second transaction *(i.e. a) Pick a winner based on some indicia of the change, such as most or least recent timestamp or sequence number, or pre-select a winner based on node information, such as the node location, node size, or node resources. b) As described above, use "relative" change information to determine the "final" value for the data that collided, and assign the final value to the data in both nodes. For example, if a part quantity initially starts at 100, and one node changes the part quantity from 100 to 30 (a net reduction of 70), while another changes the part quantity from 100 to 90 (a net reduction of 10), assigning a final value of 20 (100-70-10=100-10-70=20) to the part quantity on both nodes resolves the collision, col. 27, lines 7-21; See Fig. 8).*

Schrader teaches hierarchical data *(i.e. The database module 1407 is an interface to the user's data, which is stored in the transaction database. Records may be fetched or saved by the database module 1407 and this module allows all the other*

modules to access the transaction database. The database module 1403 preferably stores the user's data in combined relational-hierarchical data model, though other data models may also be employed. A hierarchical model is used to organize accounts per financial institution, such that all transactions for each account are stored with the account, col. 13, line 65 to col. 14, line 19).

As per claim 16, Holenstein teaches the concurrency control method of claim 8, further comprising: making the writing access with respect to a shared copy produced by copying the data in order to reflect writing accesses made by all transactions that make accesses to the data, when the first transaction is to make the writing access with respect to a copy of the data *(i.e. a) Pick a winner based on some indicia of the change, such as most or least recent timestamp or sequence number, or pre-select a winner based on node information, such as the node location, node size, or node resources. b) As described above, use "relative" change information to determine the "final" value for the data that collided, and assign the final value to the data in both nodes. For example, if a part quantity initially starts at 100, and one node changes the part quantity from 100 to 30 (a net reduction of 70), while another changes the part quantity from 100 to 90 (a net reduction of 10), assigning a final value of 20 (100-70-10=100-10-70=20) to the part quantity on both nodes resolves the collision, col. 27, lines 7-21; See Fig. 8)*; and

storing states of the shared copy at timings at which the writing accesses were made by some of the transactions that make accesses to the data; wherein the judging step obtains the second data as data obtained by reproducing a state of the data at a

timing at which the reading access is to be made by selecting one of stored states of the shared copy which is close to the state of the data at a timing at which the reading access is to be made, making the writing access that was made by the first transaction after that timing, with respect to a selected state of the shared copy, and making the writing access that was made by the second transaction according to need, and then making the reading access with respect to a reproduced state of the data *(i.e. a) Pick a winner based on some indicia of the change, such as most or least recent timestamp or sequence number, or pre-select a winner based on node information, such as the node location, node size, or node resources. b) As described above, use "relative" change information to determine the "final" value for the data that collided, and assign the final value to the data in both nodes. For example, if a part quantity initially starts at 100, and one node changes the part quantity from 100 to 30 (a net reduction of 70), while another changes the part quantity from 100 to 90 (a net reduction of 10), assigning a final value of 20 (100-70-10=100-10-70=20) to the part quantity on both nodes resolves the collision, col. 27, lines 7-21; See Fig. 8).*

Schrader teaches hierarchical data *(i.e. The database module 1407 is an interface to the user's data, which is stored in the transaction database. Records may be fetched or saved by the database module 1407 and this module allows all the other modules to access the transaction database. The database module 1403 preferably stores the user's data in combined relational-hierarchical data model, though other data models may also be employed. A hierarchical model is used to organize accounts per*

financial institution, such that all transactions for each account are stored with the account, col. 13, line 65 to col. 14, line 19).

As per claim 18, Holenstein teaches the concurrency control method of claim 1, wherein when the judging step judges that the collision will occur, the carrying out step carries out the processing for keeping those transactions that are determined according to prescribed criteria among transactions related to the collision, to wait until other transactions related to the collision are finished *(i.e. Switching Between Synchronous and Asynchronous Replication Modes. Data replication systems are normally set to operate in either a synchronous or asynchronous replication mode. Synchronous systems are prone to failure due to a disruption in communication between nodes. Accordingly, a synchronous system may be programmed to automatically revert to an asynchronous system if such a failure is detected, col. 25, lines 22-52).*

Claims 7, 14, 17 are rejected under 35 U.S.C. 103(a) as being unpatentable over Holenstein et al. (US Patent No. 7,103,586), in view of Schrader et al. (US Patent No. 5,903,881), and further in view of Peir et al. (US Patent No. 6,725,341).

As per claim 7, Holenstein teaches the concurrency control method of claim 5, wherein when there a number of shared copies that can be recorded, those shared copies which have a higher possibility of being utilized at a time of reproducing a state in which the reading access is to be made later on are recorded at a higher priority,

among the shared copies corresponding to states at times of the writing accesses with respect to the data.

Schrader teaches hierarchical data *(i.e. The database module 1407 is an interface to the user's data, which is stored in the transaction database. Records may be fetched or saved by the database module 1407 and this module allows all the other modules to access the transaction database. The database module 1403 preferably stores the user's data in combined relational-hierarchical data model, though other data models may also be employed. A hierarchical model is used to organize accounts per financial institution, such that all transactions for each account are stored with the account, col. 13, line 65 to col. 14, line 19).*

Holenstein, Schrader do not explicitly teach an upper limit.

Peir teaches an upper limit *(i.e. Such rules that restrict memory write backs to lines likely to be needed in other caches serve to further limit the number of transfers over the system memory bus to alleviate risk of flooding the bus with L1 cache write-back activity, col. 5, lines 7-16).*

It would have been obvious to one of ordinary skill of the art having the teaching of Holenstein, Schrader, and Peir at the time the invention was made to modify the system of Holenstein, Schrader to include the limitations as taught by Peir. One of ordinary skill in the art would be motivated to make this combination in order to further limit the number of transfers over the system memory bus to alleviate risk of flooding the bus with L1 cache write-back activity in view of Peir (col. 5, lines 7-16), as doing so would give the added benefit of reducing the impact of cache line invalidation on cache

efficiency in distributed cache multiprocessor computer systems as taught by Peir (col. 2, line 63 to col. 3, line 2).

As per claim 14, Holenstein teaches the concurrency control method of claim 13, wherein when there is a number of shared copies that can be recorded, those shared copies which have a higher possibility of being utilized at a time of reproducing a state in which the reading access is to be made later on are recorded at a higher priority, among the shared copies corresponding to states at times of the writing accesses with respect to the data *(i.e. Send the queued transactions that do not cause any collision to the other nodes in their order of occurrence. For example, transactions that caused collisions can be identified by comparing a unique record indicia (such as a unique primary key) across the nodes; those that were updated on multiple nodes during the asynchronous period have collided, col. 26, lines 64 to col. 27, line 3).*

Schrader teaches hierarchical data *(i.e. The database module 1407 is an interface to the user's data, which is stored in the transaction database. Records may be fetched or saved by the database module 1407 and this module allows all the other modules to access the transaction database. The database module 1403 preferably stores the user's data in combined relational-hierarchical data model, though other data models may also be employed. A hierarchical model is used to organize accounts per financial institution, such that all transactions for each account are stored with the account, col. 13, line 65 to col. 14, line 19).*

Holenstein, Schrader do not explicitly teach an upper limit.

Peir teaches an upper limit *(i.e. Such rules that restrict memory write backs to lines likely to be needed in other caches serve to further limit the number of transfers over the system memory bus to alleviate risk of flooding the bus with L1 cache write-back activity, col. 5, lines 7-16).*

It would have been obvious to one of ordinary skill of the art having the teaching of Holenstein, Schrader, and Peir at the time the invention was made to modify the system of Holenstein, Schrader to include the limitations as taught by Peir. One of ordinary skill in the art would be motivated to make this combination in order to further limit the number of transfers over the system memory bus to alleviate risk of flooding the bus with L1 cache write-back activity in view of Peir (col. 5, lines 7-16), as doing so would give the added benefit of reducing the impact of cache line invalidation on cache efficiency in distributed cache multiprocessor computer systems as taught by Peir (col. 2, line 63 to col. 3, line 2).

As per claim 17, Holenstein teaches the concurrency control method of claim 16, wherein when there is a number of shared copies that can be recorded, those shared copies which have a higher possibility of being utilized at a time of reproducing a state in which the reading access is to be made later on are recorded at a higher priority, among the shared copies corresponding to states at times of the writing accesses with respect to the data *(i.e. Send the queued transactions that do not cause any collision to the other nodes in their order of occurrence. For example, transactions that caused collisions can be identified by comparing a unique record indicia (such as a*

unique primary key) across the nodes; those that were updated on multiple nodes during the asynchronous period have collided, col. 26, lines 64 to col. 27, line 3).

Schrader teaches hierarchical data *(i.e. The database module 1407 is an interface to the user's data, which is stored in the transaction database. Records may be fetched or saved by the database module 1407 and this module allows all the other modules to access the transaction database. The database module 1403 preferably stores the user's data in combined relational-hierarchical data model, though other data models may also be employed. A hierarchical model is used to organize accounts per financial institution, such that all transactions for each account are stored with the account, col. 13, line 65 to col. 14, line 19).*

Holenstein, Schrader do not explicitly teach an upper limit.

Peir teaches an upper limit *(i.e. Such rules that restrict memory write backs to lines likely to be needed in other caches serve to further limit the number of transfers over the system memory bus to alleviate risk of flooding the bus with L1 cache write-back activity, col. 5, lines 7-16).*

It would have been obvious to one of ordinary skill of the art having the teaching of Holenstein, Schrader, and Peir at the time the invention was made to modify the system of Holenstein, Schrader to include the limitations as taught by Peir. One of ordinary skill in the art would be motivated to make this combination in order to further limit the number of transfers over the system memory bus to alleviate risk of flooding the bus with L1 cache write-back activity in view of Peir (col. 5, lines 7-16), as doing so would give the added benefit of reducing the impact of cache line invalidation on cache

efficiency in distributed cache multiprocessor computer systems as taught by Peir (col. 2, line 63 to col. 3, line 2).

Response to Arguments

With respect to claims 1, 4-21, Applicants have amended the independent claims 1, 19, 20, 21 to recite new limitations to distinguish over the cited references; however, upon further consideration, a new ground(s) of rejection is made in view of newly found prior art.

Conclusion

Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Miranda Le whose telephone number is (571) 272-4112. The examiner can normally be reached on Monday through Friday from 10:00 AM to 6:00 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, James K. Trujillo, can be reached on (571) 272-3677. The fax number to this Art Unit is (571)-273-8300.

Any inquiry of a general nature or relating to the status of this application should be directed to the Group receptionist whose telephone number is (571) 272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Miranda Le/
Primary Examiner, Art Unit 2169